NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2002; **9**:429–444 (DOI: 10.1002/nla.300)

Sparse matrix element topology with application to AMG(e) and preconditioning[‡]

Panayot S. Vassilevski[*,†]

Center for Applied Scientific Computing, UC Lawrence Livermore National Laboratory, P.O. Box 808, L-560, Livermore, CA 94551, U.S.A.

SUMMARY

This paper defines topology relations of elements treated as overlapping lists of nodes. In particular, the element topology makes use of element faces, element vertices and boundary faces which coincide with the actual (geometrical) faces, vertices and boundary faces in the case of true finite elements. The element topology is used in an agglomeration algorithm to produce agglomerated elements (a non-overlapping partition of the original elements) and their topology is then constructed, thus allowing for recursion. The main part of the algorithms is based on operations on Boolean sparse matrices and the implementation of the algorithms can utilize any available (parallel) sparse matrix format. Applications of the sparse matrix element topology to AMGe (algebraic multigrid for finite element problems), including elementwise constructions of coarse non-linear finite element operators are outlined. An algorithm to generate a block nested dissection ordering of the nodes for generally unstructured finite element meshes is given as well. The coarsening of the element topology is illustrated on a number of fine-grid unstructured triangular meshes. Published in 2002 by John Wiley & Sons, Ltd.

KEY WORDS: element topology; element agglomeration; coarse elements; sparse matrices; nested dissection ordering; unstructured meshes; algebraic multigrid

1. INTRODUCTION

Assume that we are given a sparse matrix A, e.g. coming from finite element discretization of second-order elliptic PDEs. Following the element agglomeration algorithm proposed in Jones and Vassilevski [1], provided that an initial set of elements, i.e. lists (or sets) of degrees of freedom (or nodes), is given, one can produce similar lists of agglomerated elements, i.e. a smaller set of elements, called agglomerated elements. Each agglomerated element (AE) is formed by joining a number of elements. The algorithm under consideration exploits certain topological information about the elements; namely, one appropriately defines element faces, and based on the face–face connectivity (relation) the agglomeration procedure operates. The

[*] Correspondence to: Panayot S. Vassilevski, UC Lawrence Livermore National Laboratory, L-560, P.O. Box 808, Livermore, CA 94551, U.S.A.
[†] E-mail: vassilevski1@llnl.gov
[‡] This article is a US Government work and is in the public domain in the USA

Contract/grant sponsor: U.S. Department of Energy; contract/grant number: W-7405-Eng-48.

Received 8 September 2001
Revised 15 March 2002

details will be summarized in Section 2. Typically, in practice, a finite element mesh generator will provide the initial information we need, i.e. the relation 'element_node' and also the relations 'element_face' and 'face_node'. The main point of the present paper is based on an agglomeration step which produces agglomerated elements to define their topology, for example faces of AEs (further denoted by AEfaces) and respective topological relations. In order to be able to recursively apply certain algorithms one needs to create the same information after each step of agglomeration, i.e. the new relations 'AE_AEface', 'AEface_node', etc. More specifically, the recursive application of the agglomeration algorithm from [1], which relies on the relations 'face_face', 'element_face' and 'face_element' requires that those have to be created at every coarsening step, i.e. after an agglomeration step one needs to compute the new relations 'AEface_AEface' and 'AE_AEface', 'AEface_AE'. The correct definitions of the above relations and the algorithms for their computation are given in Section 2. Equivalent definitions of these relations were proposed in Reference [1]. In the present paper, as it turns out, we were able to formulate all algorithms for generating the needed relations in terms of (sparse matrix)$\times$(sparse matrix) products and forming the transpose of sparse matrix. In most of the cases only the symbolic part of these sparse matrix operations is needed. More precisely, only the algorithm that computes faces of the agglomerated elements requires the numerical part of a sparse matrix$\times$(sparse matrix) product.

A main application of the proposed sparse matrix element topology is given in the coarse grid selection phase of AMG(e) (AMG stands for algebraic multigrid whereas AMGe stands for AMG for finite element problems) as well as in the construction of AMG(e) interpolation mappings. Some details are outlined in Section 4.

The recursive definition of coarse elements and their topology is a useful tool in several other applications; for example, the notion of element matrices on coarse levels can be useful to devise powerful (spectral) AMGe methods. In addition, the notion of vertices of the coarse elements is useful to define coarse discretizations to certain non-linear problems. Those are topics presented in greater detail elsewhere [1–4]. Here, only the modification needed for the construction of coarse non-linear finite element elliptic operators is briefly outlined in Section 5.

Another application of the sparse matrix element topology utilizes the faces as separators. That is, one can define 'boundary' of an element as the union of its faces. Similarly, to identify the interior nodes of E one uses its faces, the AEfaces. They form the boundary of E. Applying a proper two-by-two block ordering (interior and boundary nodes) of the AE nodes recursively (i.e. consecutive agglomeration and face identification of the new agglomerates), one ends up with a block nested dissection-type ordering of the set of fine degrees of freedom (nodes). Some more details are given in Section 3.

Section 6 contains some illustration of the agglomeration algorithm which exploits the sparse matrix element topology.

Finally, some conclusions are drawn at the end of the paper.

2. ELEMENT TOPOLOGY AND RELATION-BASED AGGLOMERATION ALGORITHMS

2.1. Main definitions and constructions

By definition, an element is a 'list of degrees of freedom' (or list of nodes), $e=\{d_1,\ldots,d_{n_e}\}$, and we are given an overlapping partition $\{e\}$ of $\mathscr{D}$ (the set of degrees of freedom or nodes).

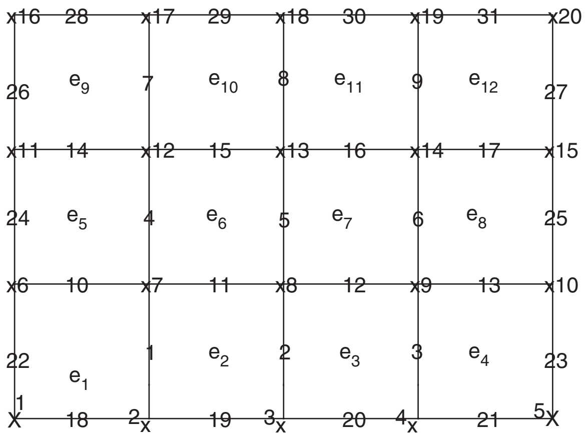

Figure 1. Sample grid: 12 elements, 31 faces and 20 nodes.

In practice, each element e is associated with an element matrix A_e, an $n_e \times n_e$ matrix, then the given sparse matrix A is assembled from the individual element matrices A_e in the usual way, i.e.

$$\mathbf{w}^{\mathrm{T}}A\mathbf{v} = \sum_e \mathbf{w}_e^{\mathrm{T}}A_e\mathbf{v}_e$$

Here, $\mathbf{v}_e = \mathbf{v}|_e$, i.e. restriction to subset ($e \subset \mathscr{D}$).

In what follows, we shall not assume explicit knowledge of the element matrices A_e, more precisely, the element matrices will be only needed in one of the applications but not in the construction of the element topology.

As an illustration, seen in Figure 1, one has the following elements as lists (or sets) of nodes:

$$\begin{aligned}
e_1 &= \{1,2,6,7\}\\
e_2 &= \{2,3,7,8\}\\
e_3 &= \{3,4,8,9\}\\
e_4 &= \{4,5,9,10\}\\
e_5 &= \{6,7,11,12\}\\
e_6 &= \{7,8,12,13\}\\
e_7 &= \{8,9,13,14\}\\
e_8 &= \{9,10,14,15\}\\
e_9 &= \{11,12,16,17\}\\
e_{10} &= \{12,13,17,18\}\\
e_{11} &= \{13,14,18,19\}\\
e_{12} &= \{14,15,19,20\}
\end{aligned}$$

Table I. Relation '**element_node**' corresponding to Figure 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1				1	1													
2		1	1				1	1												
3			1	1				1	1											
4				1	1				1	1										
5						1	1				1	1								
6							1	1				1	1							
7								1	1				1	1						
8									1	1				1	1					
9											1	1				1	1			
10												1	1				1	1		
11													1	1				1	1	
12														1	1				1	1

Assume that the following relation (in the sense of [5]) '**element_node**' is given; that is, the incidence 'element' i (rows) contains 'node' j (columns), i.e. '**element_node**' can be viewed as the rectangular (Boolean) sparse matrix of ones in the (i,j)-position if element i contains node j and zeros elsewhere. The size of the matrix is (number of elements)$\times$(number of nodes).

The relation '**element_node**' corresponding to Figure 1 is shown in Table I. The incidence 'node' i belongs to 'element' j, is simply given by the transpose of the above rectangular sparse matrix, i.e. **node_element** $=$ (**element_node**)^T.

One can consider a number of useful relations (easily computable as operations between sparse matrices):

'**element_element**' $=$ '**element_node**' $\times$ '**node_element**',

'**node_node**' $=$ '**node_element**' $\times$ '**element_node**'.

The first one shows the incidence 'element' i intersects 'element' j, that is, the (i,j) entry of the '**element_element**' is one if 'element' i and 'element' j have a common node, otherwise the entry is zero.

The second relation ('**node_node**') shows the sparsity pattern of the (assembled) finite element matrix $A=(a_{ij})$. This is seen as follows. The non-zero entries (i,j) of '**node_node**' show that 'node' i is connected to 'node' j in the sense they belong to a common element. Hence, the corresponding entry $a_{i,j}$ of A is possibly non-zero. This is exactly the case since $a_{i,j}$ can be non-zero only if the nodes i and j belong to the same element. Here, we assume that each node represents a degree a freedom, that is, it is associated with a finite element basis function whose support is contained in the union of elements sharing that node.

The relation '**node_node**' corresponding to Figure 1 is illustrated in Table II.

In practice, one can implement these relations using any available sparse matrix format, for example CSR (compressed sparse row) format. For parallel implementation, one has to use appropriate parallel sparse matrix format.

2.2. Element faces

In practice, it is typical that a finite element mesh generator can provide the fine-grid element topology, namely the relations

'**element_face**', '**face_element**', '**face_node**', '**face_face**', etc.

Table II. Relation '**node_node**' corresponding to Figure 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1				1	1													
2	1	1	1			1	1	1												
3		1	1	1			1	1	1											
4			1	1	1			1	1	1										
5				1	1				1	1										
6	1	1				1	1				1	1								
7	1	1	1			1	1	1			1	1	1							
8		1	1	1			1	1	1			1	1	1						
9			1	1	1			1	1	1			1	1	1					
10				1	1				1	1				1	1					
11						1	1				1	1				1	1			
12						1	1	1			1	1	1			1	1	1		
13							1	1	1			1	1	1			1	1	1	
14								1	1	1			1	1	1			1	1	1
15									1	1				1	1				1	1
16											1	1				1	1			
17											1	1	1			1	1	1		
18												1	1	1			1	1	1	
19													1	1	1			1	1	1
20														1	1				1	1

Table III. Relation '**boundarysurface_node**' corresponding to Figure 1. Boundary surface 1 is the left vertical, boundary surface 2 is the bottom horizontal, boundary surface 3 is right vertical and boundary surface 4 is the top horizontal.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1					1					1					1				
2	1	1	1	1	1															
3					1					1					1					1
4																1	1	1	1	1

If the initial set of element faces is not given one can define a '**face**' (as a list of nodes) as a maximal intersection set. Recall that every element is a list (set) of nodes. Consider all pairwise intersections of elements, $e \cap e_1$, $e_1 \neq e$. Then all maximal sets form the faces of e. Here 'maximal' stands for a set which is not a proper subset of any other intersection set. The above definition will only give the set of interior faces. One may assume that additional information about the domain boundary is given in terms of lists of nodes called boundary surfaces. Then, a face is a maximal intersection set of the previous type, or a maximal intersection set of the type $e \cap$ 'boundary surface'.

In Figure 1 one can define four boundary surfaces and can construct the relation '**boundarysurface_node**' shown in Table III.

At any rate, we assume that the faces of the initial set of elements are given either by a mesh generator or they can be computed as the maximal intersection sets. I.e. we assume that the relations '**element_face**' and '**face_node**' are given.

One can then construct, based on sparse matrix manipulations, the following relations: '**face_element**' = ('**element_face**')$^{\mathrm{T}}$, '**node_face**' = ('**face_node**')$^{\mathrm{T}}$ and '**face_face**' = '**face_node**' $\times$ '**node_face**'.

2.3. *Element agglomeration*

The topological information can be used to devise an algorithm to agglomerate elements, i.e. to construct a new overlapping partition $\{E\}$ of $\mathscr{D}$ where each $E = e_1 \cup e_2 \cdots \cup e_p$. In other words, to build the new relation '**AE_element**' where **AE** stands for 'agglomerated element'.

The following algorithm has been proposed in Reference [1]. The motivation was to produce 'quasiuniform' 'AEs'. In particular, this algorithm restores coarse rectangular or triangular elements (up to boundary effects). Since the algorithm, based on the topological relations, assigns appropriate weights (higher weights for higher dimensional relation) elements that share a face are more likely to be agglomerated than elements that share only a node. This should intuitively lead to more compact agglomerated elements.

Algorithm 2.1 (*Agglomeration of elements*)
Given the relations '**face_face**', '**element_face**' and '**face_element**', and a weight function $w = w(f)$ (f is a face) initially set to zero. One performs the following steps to generate 'AEs':

1. find a face f with maximal $w(f) \geqslant 0$, then set $w(f) = -1$ and add on the list of the current 'AE' the elements e_1 and e_2 which form f, i.e. $f = e_1 \cap e_2$,
2. update $w(g)$ for all faces g connected to f (based on the relation '**face_face**'), according to the following topological rule, $w(g) := w(g) + 1$ if g is connected to f and once more $w(g) := w(g) + 1$ if g and f belong to a same element (here one uses the relation '**face_element**');
3. if for all faces g of the already agglomerated elements e in the current 'AE' the weight $w(g)$ is less than $w(f)$ where f was the last eliminated face, the agglomeration procedure for the current 'AE' is terminated. (Here one uses the relation '**element_face**'.) Label the faces g of the elements in 'AE' as eliminated, i.e. set $w(g) = -1$. Then, go to step 1 to build a new 'AE' or stop if all faces have already been eliminated, i.e. if $w(f) = -1$ for all f.

The above algorithm can be efficiently implemented based on linked lists, that is, one can form a linked list of the faces ordered according to their weight and operate on that list based on their changing weight. One removes a face from the list if its weight becomes -1 or rearranges the list according to the changes of the weight of the faces occurring step (2) of the algorithm.

2.4. *Faces of AEs*

The purpose of constructing AEs is to define similar topological relations for them and perform further agglomeration steps by recursion. For this reason, we have to be able to define faces of AEs which we will call 'AEfaces'. Assume that the relation '**AE_element**' has been constructed by Algorithm 2.1, then one can build the relation (as a Boolean sparse matrix) '**AE_face**' = '**AE_element**' $\times$ '**element_face**'. This represents the AEs in terms of the faces of the original elements. The idea is that every two AEs that share a face of the original elements should share 'AEface' as well. That is, one can define faces of agglomerated elements, 'AEface's, based on '**AE_face**' by simply intersecting the lists (sets) of every two AEs that share a common face, or if the relation '**boundarysurface_face**' is given, by intersecting every AE with a boundary surface if they share a common face of the original elements. By doing so (intersecting two different AEs in terms of faces, or intersecting an AE in terms of faces and a boundary surface also in terms of faces), one gets the 'AEface's of the 'AEs' in terms of the faces of the original elements. Thus, one constructs the new relations '**AEface_face**', and '**AE_AEface**'. The above definition of the (interior) AEfaces can be formalized in the following algorithm.

Algorithm 2.2 (*Creating interior AEfaces*)
Given the relations,

'**AE_element**', '**element_face**'

implemented as Boolean sparse matrices. In order to produce as an output the new relations

'**AEface_AE**' and '**AEface_face**'

one performs the following steps:

- form the relations:

  1. $$\text{'}\mathbf{AE\_face}\text{'} = \text{'}\mathbf{AE\_element}\text{'} \times \text{'}\mathbf{element\_face}\text{'}$$

  2. $$\text{'}\mathbf{AE\_AE}\text{'} = \text{'}\mathbf{AE\_face}\text{'} \times (\text{'}\mathbf{AE\_face}\text{'})^{\mathrm{T}}$$

- assign an '**AEface**' to each (undirected) pair ($\mathbf{AE}_1$, $\mathbf{AE}_2$) of different AEs from the relation '**AE_AE**'. The new relation '**AEface_AE**' is stored also as a Boolean rectangular sparse matrix.
- form the product (including the numerical part of the sparse matrix–matrix multiply):

$$\text{'}\mathbf{AEface\_AE\_face}\text{'} \equiv \text{'}\mathbf{AEface\_AE}\text{'} \times \text{'}\mathbf{AE\_face}\text{'}$$

- finally, the required relation

  '**AEface_face**'

  is obtained by deleting all entries of '**AEface_AE_face**' with numerical value 1.

The last step of the above algorithm is motivated as follows. The non-zero entries of the sparse matrix '**AEface_AE_face**' are either 1 or 2 (since a face can belong to at most two AEs). An entry a_{ij} of '**AEface_AE_face**' with value 2 indicates that the '**AEface**' corresponding to the row index 'i' of a_{ij} has a face corresponding to the column index 'j' with a weight 2. This means that the face 'j' is common to the two AEs which define the AEface 'i'. Therefore the face 'j' 'belongs' to the AEface 'i' (since it is a shared face by the two neighbouring AEs which form the AEface 'i'). The entries a_{ij} of '**AEface_AE_face**' with value one correspond to a face 'j' which is interior to one of the AEs (from the undirected pair of AEs that forms the AEface 'i') and hence is of no interest here.

Remark 2.1
If the relation '**boundarysurface_face**' is given one can use it to define boundary AEfaces. One first forms the relation '**AE_boundarysurface**' = '**AE_face**' $\times$ ('**boundarysurface_face**')^T and then to each AE which is connected to a boundary surface (that is, to each pair (AE, boundarysurface) from the relation '**AE_boundarysurface**') one assigns (a boundary) AEface. Thus, the relation '**AE_AEface**' obtained from Algorithm 2.2 is augmented with the boundary AEfaces. The list '**AEface_face**' is augmented with the intersection sets ('**AE_face**') $\cap$ ('**boundarysurface_face**') for every related pair (AE, boundarysurface) from the relation '**AE_boundarysurface**'. This means that we intersect every row of '**AE_face**' with any related to it row of '**boundarysurface_face**'.

Thus the following information needed for the next agglomeration step is created, that is, we have built the new ('coarse') relations, '**element_face**' $\equiv$ '**AE_AEface**', '**face_element**' $\equiv$ '**AEface_AE**' = ('**AE_AEface**')^T.

The last relation needed in Algorithm 2.1 is '**face_face**' $\equiv$ '**AEface_AEface**'. A proper way to define '**AEface_AEface**' is as the following triple product:

$$\text{'}\mathbf{AEface\_AEface}\text{'} = \text{'}\mathbf{AEface\_face}\text{'} \times \text{'}\mathbf{face\_face}\text{'} \times (\text{'}\mathbf{AEface\_face}\text{'})^T$$

This definition (of coarse '**face_face**' relation) does not make use of the relation '**face_node**', i.e. no node knowledge is required in the recursive application of Algorithm 2.1. The same holds for Algorithm 2.2.

The new (coarse) relation '**boundarysurface_face**' $\equiv$ '**boundarysurface_AEface**' is readily constructed as the product '**boundarysurface_face**' $\times$ ('**AEface_face**')^T.

If nodal relations are needed on coarse levels, one can easily construct them. For example, the new (coarse) relation '**face_node**' $\equiv$ '**AEface_node**' is computed as the product '**AEface_node**' = '**AEface_face**' $\times$ '**face_node**' assuming that the fine relation '**face_node**' was given.

3. NESTED DISSECTION ORDERING

We now adopt a dual notation. First, we consider any given relation '**object1_object2**' as a rectangular Boolean sparse matrix, and second, each row of this matrix gives a set of '**object2**s', that is, the rows '**object1**' can be considered as sets consisting of '**object2**' entries. Hence, we can operate with these rows as sets and in particular we can find their intersection and union. We will in particular view a relation '**object1_object2**' as the set obtained by the union of its rows.

Assume now that one has generated a sequence of agglomerated elements and their topology. In particular, we need $\{(\text{'}\mathbf{face\_node}\text{'})_k\}$, and $\{(\text{'}\mathbf{AEface\_face}\text{'})_k\}$, $k \geqslant 0$. (For convenience, we let ('**AEface_face**')$_0$ be the identity Boolean matrix, i.e. at the initial fine level $k = 0$ 'AEface' equals 'face'. Similarly, for other purposes, it is also convenient to let ('**AE_element**')$_0$ be the identity relation, that is 'AE' equal 'element' on the initial level.)

Having the topological information at fine level $k=0$, in addition to the nodal information (‘**face_node**’)$_0$, one first creates the topological information recursively, in particular, one creates $\{(\text{‘}\mathbf{AEface\_face}\text{’})_k\}$, $k \geqslant 0$. and then, by definition, one sets (‘**face_node**’)$_k$ = (‘**AEface_face**’)$_k \times$ (‘**face_node**’)$_{k-1}$ for $k > 0$.

Note that, by construction, (‘**face_node**’)$_k \subset$ (‘**face_node**’)$_{k-1}$. This means that each coarse face (i.e. a face at the coarse level k) contains nodes only from the fine level $k-1$ faces.

Definition 3.1
The splitting,

- $\mathscr{S}_0 \equiv \mathscr{D} \backslash$(‘**face_node**’)$_0$;
- and for $k > 0$, $\mathscr{S}_k \equiv$ (‘**face_node**’)$_{k-1} \backslash$(‘**face_node**’)$_k$,

provides a direct decomposition of the original set of nodes $\mathscr{D}$.

In the case of regular refinement (elements of fine level $k-1$ are obtained by geometrical refinement of coarse level k elements) the above splitting gives rise to the so-called nested dissection ordering (cf., e.g. [10, Chapter 8]). Thus, in a general unstructured grid case our sparse matrix element topology leads to the following natural extension:

Definition 3.2 (*Nested dissection ordering*)
Consider the sets $\mathscr{S}_k$ defined in Definition 3.1. The splitting

$$\mathscr{D} = \bigcup_{k \geqslant 0} \mathscr{S}_k \tag{1}$$

gives rise to a block ordering of the assembled sparse matrix A (or of the relation ‘**node_node**’) called nested dissection ordering.

Two examples of a sparsity pattern of the fine-grid assembled matrix in the nested dissection ordering are shown in Figures 2 and 3. The fine-grid is similar to the one shown in Figures 4 and 5.

Nested dissection ordering is useful in certain approximate factorization algorithms, and hence can be useful in building preconditioners to A. Combined with, say, minimum degree ordering within each block of A resulting from the above-described nested dissection ordering, one may build various ILU-type factorization preconditioners (which are of multilevel type) as an alternative to the ones proposed in Reference [7]. In Reference [7] the multilevel two-by-two block-structure of the original matrix was obtained based on a reverse Cuthill–McKee algorithm. Here, we suggest as an alternative to get multilevel hierarchy by the nested dissection ordering.

4. COARSE GRID SELECTION AND BUILDING INTERPOLATION MAPPINGS IN AMG(e)

Assume that an element agglomeration step has been performed and the respective AE topology has been created. Assume also that the nodal relations have been computed. In particular, we need the relations ‘**AEface_node**’ and ‘**AE_node**’ and their derivatives.

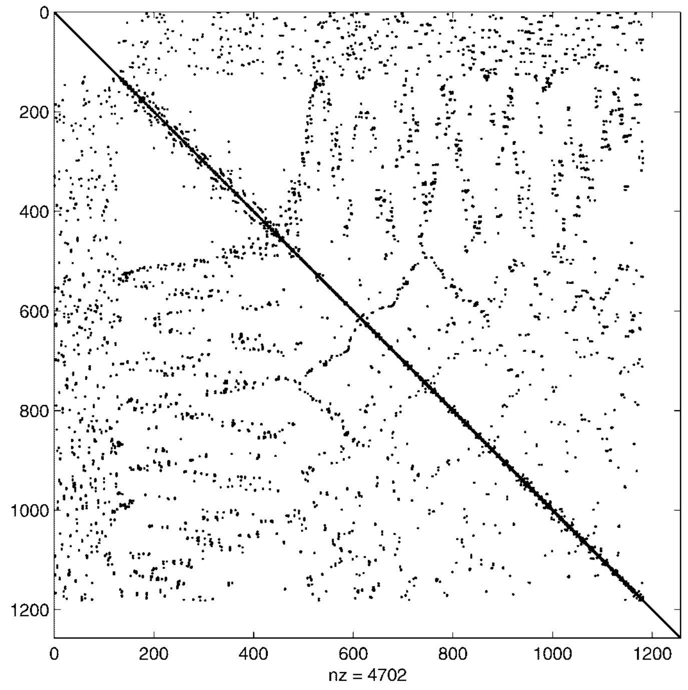

Figure 2. Typical sparsity pattern in the nested dissection ordering.

In Reference [1], the so-called set of 'vertices' has been selected as the coarse grid $\mathcal{D}_c$. One may define vertices in several ways; for example, a node is called a 'vertex' if it belongs to more than one AEface. This is easily checked based on the relation '**node_AEface**' = ('**AEface_node**')^T. Here, we assume 2-D fine-grid elements. In 3-D with this definition we will actually select as 'vertices' all nodes which are on the (geometric) edges of the AEs. We have not actually introduced 'edges' in our topological relations, but those are similarly defined as maximal intersection sets from the relations '**AE_edge**' = '**AE_element**' $\times$ '**element_edge**' assuming that on the fine grid the relation '**element_edge**' has been given (constructed).

A 'dimension-free' definition of (true) vertices was given in Reference [1]. Consider the intersection sets $I(x) = \bigcap\{\textbf{AEface}\colon\ x \in \textbf{AEface}\}$ for all $x \in \mathcal{D}$. A vertex is a minimal set $I(x)$ in the sense that $I(x)$ does not contain as a proper subset any other $I(y)$. By this construction of vertices it is clear that any AEface has at least one vertex and the same holds for any AE, i.e. any AE has at least one vertex. That is, the set of vertices provides a sort of maximal independent set. To compute minimal intersection sets is certainly more expensive than simply checking if a node belongs to more than one AEface. That is why in practice, we prefer to use the first definition (which in 3-D labels all nodes on AEedges as vertices).

Definition 4.1 (*Coarse nodes*)
A minimal set of coarse nodes $\mathcal{D}_c \subset \mathcal{D}$ is provided by the vertices of 'AEs'. Thus one has formed the relation '**node_coarsenode**'.

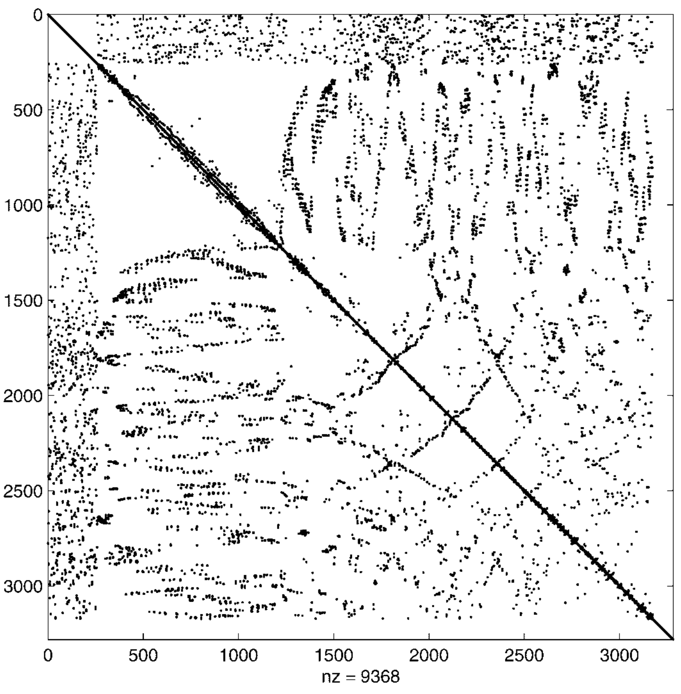

Figure 3. Typical sparsity pattern in the nested dissection ordering.

The set of vertices can be used as minimal coarse grid in AMG. (This was the choice made in Reference [1].) Then any fine node has a coarse neighbour in its neighbourhood defined by the set of AEs which contain that node. Similarly, every node on a AEface has at least one coarse node neighbour.

From complexity point of view in 3-D it is okay to select all nodes on an AEedge as coarse nodes (in the case of model uniform refinement in 3-D one can verify this).

Having '**node_coarsenode**' constructed, one can coarsen the nodal information by building

$$\text{'}\mathbf{coarseelement\_coarsenode}\text{'} = \text{'}\mathbf{AE\_node}\text{'} \times \text{'}\mathbf{node\_coarsenode}\text{'}$$

$$\text{'}\mathbf{coarseface\_coarsenode}\text{'} = \text{'}\mathbf{AEface\_node}\text{'} \times \text{'}\mathbf{node\_coarsenode}\text{'}$$

Here 'coarseface' stands for a face of the coarse elements, that is, coming from the relation '**coarseelement_coarseface**' ≡ '**AE_AEface**'.

Another main ingredient in AMG(e) is the construction of the interpolation mapping P. Then as a rule (in the symmetric positive definite case) the coarse matrix $A_c = P^T A P$.

To be more specific we will now outline one way of building interpolation matrix P : $\mathscr{V}(\mathscr{D}_c) \mapsto \mathscr{V}(\mathscr{D})$. Here, $V_c \equiv \mathscr{V}(\mathscr{D}_c)$ stands for the vector space of discrete functions (or vectors) defined on the coarse level grid $\mathscr{D}_c$.

Assume that a relation '**node_coarsenode**' has been constructed. The set of coarse nodes $\mathscr{D}_c$ does not have (here) to be a subset of $\mathscr{D}$. However, we assume that each coarse node is uniquely associated with either a vertex, or the interior of an AEface or the interior of an

AE. With some abuse of terminology we will simply say that a coarse node is either a vertex node or an AEface interior node or an AE interior node.

We assume that there is an initial imbedding mapping $Q:\mathscr{V}(\mathscr{D}_c)\mapsto\mathscr{V}(\mathscr{D})$. For example, the vertex coarse nodes can be identified as the vertex fine-grid nodes. This means that the corresponding column of Q is the unit vector with value one at the fine-grid vertex position. Assume also that AEface interior coarse nodes and AE interior coarse nodes can be imbedded into AE fine nodes (not necessarily as identity). That is, for a given AE E and a given coarse vector for a vector $\mathbf{v}_c$ supported in E (that is its non-zero entries are associated with vertices, AEface interior and AE interior all on E), then $Q_E\mathbf{v}_c$ stands for the restriction of $Q\mathbf{v}_c$ to E.

Partition the fine-grid matrix A as

$$A=\begin{bmatrix} A_{\text{int},\text{int}} & A_{\text{int},b} & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}\begin{array}{l} \}\ \text{AE interior nodes} \\ \}\ \text{AE boundary nodes} \\ \}\ \text{nodes outside AE} \end{array}$$

Let $A_{\text{int},E}$ be the rectangular matrix which maps the nodes on E into the interior of E, that is, $A_{\text{int},E}$ is the first block-row of A in the above block partitioning of A. Let $Q_{b,E}$ be the block of Q_E such that $Q_{b,E}\mathbf{v}_c=Q_e\mathbf{v}_c|_{\text{boundary of } E}$. It is clear then that $\mathbf{v}_E=P_E\mathbf{v}_c$, where

$$P_E=Q_E+\begin{bmatrix} -(A_{\text{int},\text{int}})^{-1}A_{\text{int},E}Q_E \\ 0 \end{bmatrix}\begin{array}{l} \}\ \text{AE interior fine nodes} \\ \}\ \text{AE boundary nodes} \end{array}$$

solves the constrained minimization problem

$$(\mathbf{v}_E)^{\mathrm{T}}A_E\mathbf{v}_E\mapsto\min\quad\text{over}\quad(\mathbf{v}_E)|_{\text{boundary of } E}=(Q_E\mathbf{v}_c)|_{\text{boundary of } E}$$

Here A_E is the principal part of A corresponding to E. We have also assumed that A, hence A_E, is symmetric positive definite.

One can come up with various interpolation rules P_E by choosing specific imbedding mappings Q_E of the coarse nodes into the AE fine nodes. A fairly general scheme to build interpolation matrices was presented in Reference [8] which generalizes the classical AMG interpolation rules (from Reference [9], see also Reference [10] and earlier in Reference [11]) as well as some more recent ones proposed in Reference [12]. Other approaches, exploiting energy minimization principle to build a coarse space are found in References [13, 14]. If some additional information is assumed to be available, like element matrices (as in References [15, 1]) one can build interpolation rules again based on local energy minimization principle. Other approaches exploit the null-space modes called 'rigid body motions' to be exactly represented on coarse levels (see, Reference [16]). Geometrical information (like specific type of finite element mesh, e.g. triangular mesh) can be utilized in the coarse node selection phase, as in Reference [17], or even without explicit additional information as in Reference [18] based only on matrix graph connectivity.

An important point is that if one selects the imbedding Q_E such that AEface coarse nodes are imbedded only into AEface interior fine nodes (on the same AEface) and AE interior coarse nodes are imbedded into AE interior fine nodes (in the same AE) then $Q_{b,E}$ will depend only on the AEfaces (hence will remain the same on the AEface independently of the AE which shares that AEface). It is clear then that the local mappings P_E can be used to define coarse element matrices. That is, $A^{c}_{E_c}=(P_E)^{\mathrm{T}}A_EP_E$ is the actual coarse element matrix, where A_E is (now) the assembled matrix corresponding to the agglomerate E from the fine grid element matrices $\{A_e,\ e \subset E\}$. $A_c = P^T A P$ can be assembled in the usual way from the $\{A^c_{E_c}\}$. Here E_c is the set of coarse nodes associated with E. More details about preserving the notion of coarse element matrices are found in Reference [1], see also Reference [3].

In some cases, one may not be satisfied with the quality of the resulting coarse grids; therefore richer sets can be used to define $\mathscr{D}_c$. An extreme choice is to let all nodes on the AEfaces, form $\mathscr{D}_c$. And in order to keep the sparsity of the resulting coarse matrix somewhat under control, one may allow some nodes in the interior of the 'AEs' to be coarse. The choice of such nodes can be determined adaptively, in order to make the A_{ff} block which corresponds to the 'fine node'-'fine node' connections in the fine-grid matrix be better conditioned. In Reference [12] such a procedure was called 'compatible relaxation'. Alternatively, one may select other degrees of freedom as coarse nodes, such as, for example, certain sets of eigenvectors corresponding to a lower part of the spectrum of Schur complements of neighbourhood matrices associated with AE interior nodes and with AEface interior nodes (as chosen in Reference [3]). But we shall not go into more details here since this is not the main topic of the present paper.

5. APPLICATION TO NON-LINEAR ELLIPTIC FINITE ELEMENT PROBLEMS

Having the ability to generate a sequence of coarse triangulations, and define 'vertices', 'elements' and 'element matrices' on all coarse levels by algebraic means, one can easily come up with meaningful coarse non-linear operators based only on a fine-grid finite element non-linear discretization problem.

Consider, for any two admissible functions u and φ, the non-linear elliptic operator $\mathscr{L}$ for given non-linear positive coefficients $a = a(u)$ and $g = g(u)$,

$$(\mathscr{L}(u)u, \varphi) = \sum_T \left[a_T(u) \int_T \nabla u \cdot \nabla \varphi \,\mathrm{d}x + g_T(u) \int_T u\varphi \,\mathrm{d}x \right]$$

Here, T runs over the set of elements from a given triangulation of the problem domain, and

$$a_T(u) = a\left(\frac{1}{\text{number of vertices of } T} \sum_{x_v - \text{vertex of element } T} u(x_v) \right) \tag{2}$$

$g_T(u)$ is analogously defined.

It is clear, for finite element functions u and φ, that the first expression can be rewritten in terms of their coefficient vectors $\mathbf{u}$ and $\underline{\varphi}$ as

$$(\mathscr{L}(u)u, \varphi) = \sum_T [a_T(u)(\underline{\varphi}_T)^T A_T \mathbf{u}_T + g_T(u)(\underline{\varphi}_T)^T G_T \mathbf{u}_T] \tag{3}$$

where A_T is the element matrix for the Laplacian, G_T is the element mass matrix and $\mathbf{v}_T = \mathbf{v}|_T$ is the restriction of a given vector $\mathbf{v}$ on the set of nodes (degrees of freedom) in T.

The same expression (3) can be used on coarse triangulations to define coarse non-linear operators. To do that one uses coarse elements obtained by agglomeration and coarse element matrices computed as in the AMGe method. That is, assuming we have access to the fine-grid element matrices for both the Laplacian and the identity operator (leading to mass matrices) one can compute their coarse counterparts on all coarse levels.

The averaging formula (2) makes sense on coarse levels since we have the notion of 'coarse element vertices'. More general (and more accurate) averaging is also possible.

More details in this direction are found in Reference [4].

6. ILLUSTRATION OF AGGLOMERATED UNSTRUCTURED FINITE ELEMENT MESH

Finally, in this section, we present a typical set of agglomerated elements produced by the topological agglomeration Algorithm 2.1 in Figures 4 and 5.

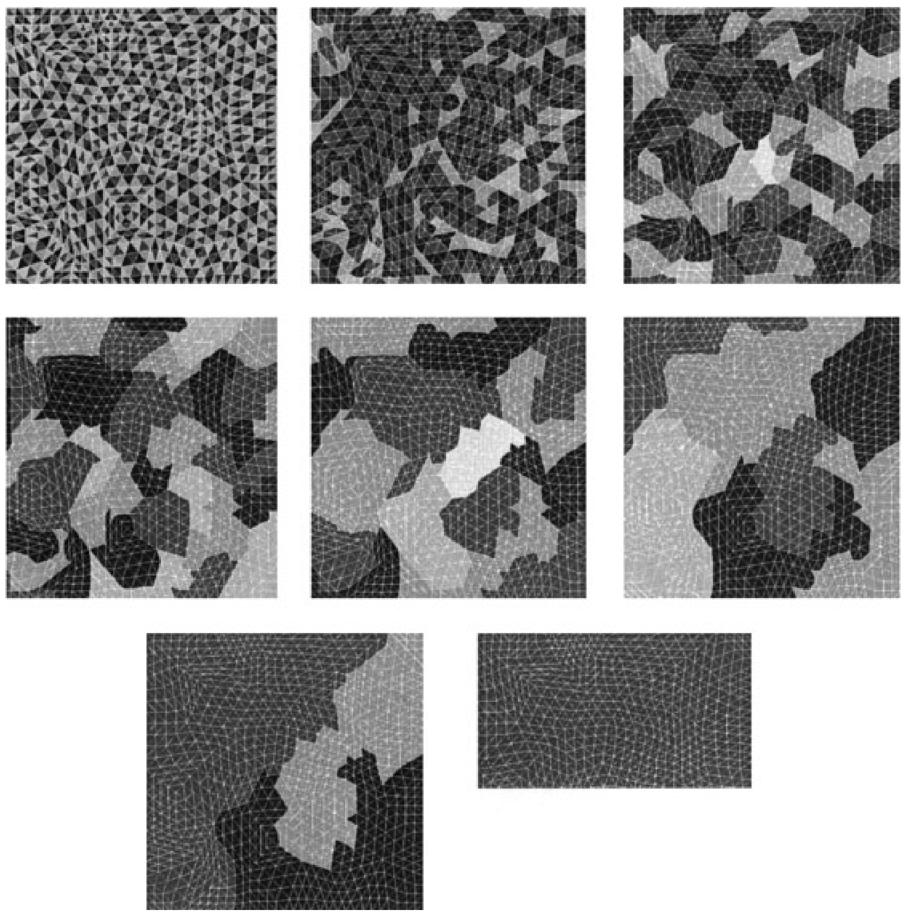

Figure 4.

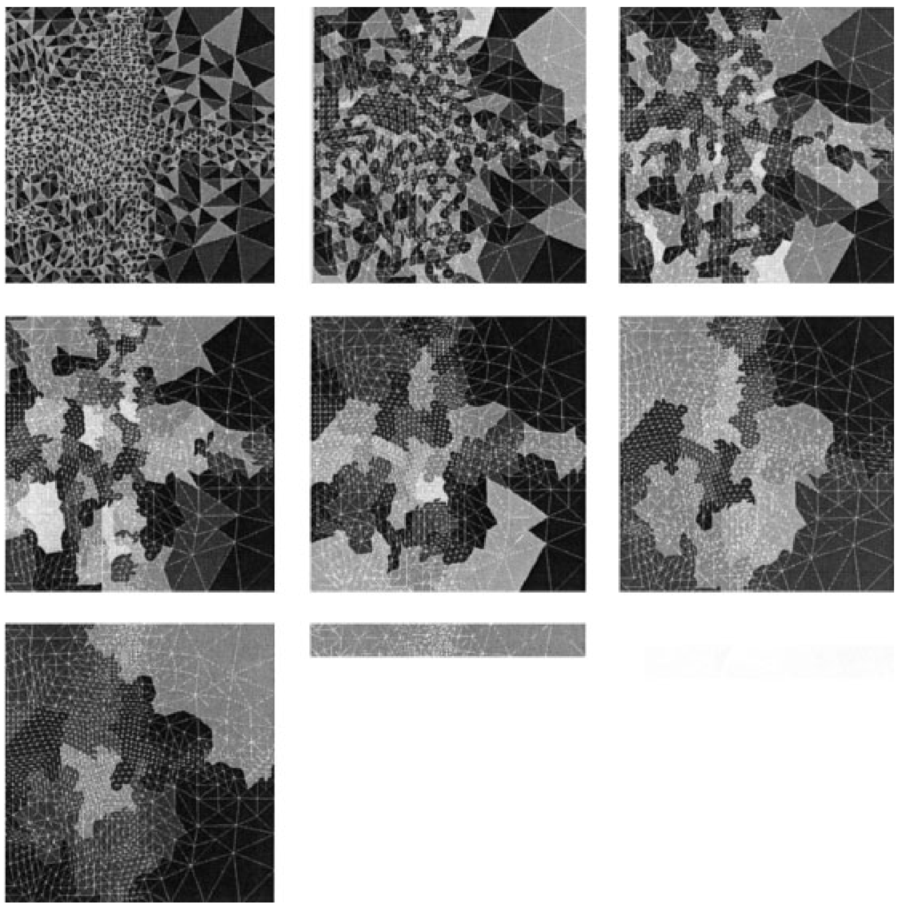

Figure 5.

7. CONCLUSIONS

A general conclusion is that the finite elements and element matrices provide additional useful information (compared to the assembled matrix only) and one can take advantage of it. More specifically,

- Topological relations implemented as Boolean sparse matrices are useful tools in dealing with finite element problems on unstructured meshes; in particular, they are useful to agglomerate and coarsen fine elements with target application the (spectral) AMGe. In this respect,

  1. we have provided definitions and algorithms to coarsen elements and their topology; they rely on sparse matrix transpose and matrix–matrix product operations;
  2. the coarse element topology can be used to define minimal coarse grids (based on vertices of AEs);

3. if one has a parallel agglomeration algorithm the rest of the AMG(e) algorithms is conceptually straightforward parallelizable; here one needs to choose a parallel sparse matrix storage and to implement sparse matrix transpose and matrix–matrix product operations in parallel;

- The derivation of coarse finite element discretizations of non-linear second-order elliptic problems is straightforward.
- To reorder the matrix in a block nested dissection ordering is also straightforward.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

REFERENCES

1. Jones JE, Vassilevski PS. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing* 2001; **23**:109–133.
2. Chartier T, Falgout R, Henson V, Jones J, Manteuffel T, McCormick S, Ruge J, Vassilevski PS. Spectral AMGe. Preprint 2001.
3. Chartier T, Falgout R, Henson V, Jones J, Manteuffel T, McCormick S, Ruge J, Vassilevski PS. Spectral Agglomeration AMGe (in preparation).
4. Jones JE, Vassilevski PS, Woodward CS. AMGe coarsening for non-linear elliptic problems (in preparation).
5. Cormen TH, Leiserson CE, Rivert RL. *Introduction to Algorithms*. MIT Press: Cambridge, Massachusetts, London, England, 1999.
6. George A, Liu JW. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Inc.: Englewood Cliffs, NJ, 1981.
7. Bank RE, Smith RK. An algebraic multilevel multigraph algorithm. *SIAM Journal on Scientific Computing* 2002; **23**:1572–1592.
8. Henson VE, Vassilevski PS. Element-free AMGe: general algorithms for computing interpolation weights. *SIAM Journal on Scientific Computing* 2001; **23**:629–650.
9. Ruge JW, Stüben K. Algebraic multigrid. In: *Multigrid Methods*, McCormick S (ed.). Philadelphia, PA, 1987; 73–130.
10. Stüben K. Algebraic Multigrid (AMG): an introduction with applications. *GMD Report* 53, GMD—Forschungszentrum Informationstechnik GmbH, Schloss Birlinghoven, Sankt Augustin, Germany, 1999.
11. Brandt A, McCormick S, Ruge JW. Algebraic multigrid (AMG) for sparse equations. In *Sparsity and its Applications* (*Loughborough 1983*), Evans DJ (ed.). Cambridge University Press: Cambridge, 1985; 257–284.
12. Brandt A. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis* 2002; **10**:1–20.
13. Wan WL, Chan TF, Smith BF. An energy minimizing interpolation for robust multigrid methods. *SIAM Journal on Scientific Computing* 2000; **21**:1632–1649.
14. Mandel J, Brezina M, Vanek P. Energy optimization of algebraic multigrid bases. *Computing* 1999; **62**:205–228.
15. Brezina M, Cleary AJ, Falgout RD, Henson VE, Jones JE, Manteuffel TA, McCormick SF, Ruge JW. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal on Scientific Computing* 2000; **22**:1570–1592.
16. Vanek P, Mandel J, Brezina M. Algebraic multigrid by smoothed aggregation for second order and fourth order elliptic problems. *Computing* 1996; **56**:179–196.
17. Chan TF, Xu J, Zikatanov LT. An agglomeration multigrid method for unstructured grids. In *Proceedings of the 10th International Conference on Domain Decomposition Methods*, August 10–14, 1977, Boulder, CO, Mandel J, Farhat C, Cai X-C. (eds). Amer. Math. Soc.: Providence, RI, 1998; 67–81.
18. Chan TF, Xu J, Zikatanov LT. A multigrid method based on agglomerated coarse spaces. Preprint, 1999.